Problem 1. The time complexity of Prim's algorithm is the following:

$$\text{Time Complexity} = \Theta(V) \cdot T_{EXTRACT-MIN} + \Theta(E) \cdot T_{DECREASE-KEY} \tag{1}$$

If all edge weight in a graph are integers in the range $[1, |V|]$, it does not change anything, because such restriction does not affect on $T_{EXTRACT-MIN}$, and $T_{DECREASE-KEY}$ of array, binary heap and Fibonacci heap. Therefore, using Fibonacci heap takes the minimum running time: $O(E + V \lg V)$.
However, if they are integers in the range in $[1, W]$ for some constant W, we can use array as data structure, which $T_{EXTRACT-MIN} = O(W) = O(1)$, and $T_{DECREASE-KEY} = O(1)$. Therefore, time complexity will be $\Theta(V + E)$

Problem 2. Finding the reliable path between s and t is that finding a path $P = (s, v_{i_1}, \cdots, v_{i_k}, t)$ which has maximum values $v(P)$, where

$$v(P) = r(s, v_{i_1}) \times r(v_{i_2}, v_{i_3}) \times \cdots \times r(v_{i_k}, t) \tag{2}$$

Modify the equation by taking logarithm and multiply -1.

$$-\lg v(P) = -(\lg r(s, v_{i_1}) + r(v_{i_2}, v_{i_3}) + \cdots + r(v_{i_k}, t)) \tag{3}$$

Then, let $r'(u, v) = -\lg r(u, v)$

$$-\lg v(P) = r'(s, v_{i_1}) + r'(v_{i_2}, v_{i_3}) + \cdots + r'(v_{i_k}, t) \tag{4}$$

Since maximizing $v(P)$ is equal as minimizing $-\lg v(P)$, a most reliable path P can be obtained by finding shortest path on graph $G' = (V, E, r')$ from s to t. Since for every u, v, $0 \le r(u, v) \le 1$, so $0 \le r'(u, v)$, which means there are no negative weight edges. Therefore, Dijkstra algorithm can be used to find a shortest path.

Problem 3. The goal of the problem is finding $c_{i_1}, \cdots, c_{i_k}$ such that

$$R[i_1, i_2] \times \cdots \times R[i_k, i_1] > 1 \tag{5}$$

Modify the equation by taking logarithm on both side and multiply -1.

$$-\lg R[i_1, i_2] - \cdots - \lg R'[i_k, i_1] < 0 \tag{6}$$

Then, let $R'[i, j] = -\lg R[i, j]$.

$$R'[i_1, i_2] + \cdots + R'[i_k, i_1] < 0 \tag{7}$$

Therefore, the sequence of countries $c_{i_1}, \cdots, c_{i_k}$ can be obtained by finding negative weight cycle on $G = (V, E, w)$ where $V = \{c_i\}$, $E = V \times V$ and $w(i, j) = -\ln R[i, j]$. Bellman-Ford algorithm can be applied to detect negative weight cycle, that runs in $O(VE)$. A following simple patch could be applied to find negative weight cycle, if it exists.
(0) In the initalization step, make $|V| \times |V|$ 2D array.
(1) In the relaxation step, store predessor(u) in i th row of the array if relaxation occurs.
(2) In the final check step, if $d[v] > d[u] + w(u, v)$, instead of report the existance, simply trace predessor from v until i) return to v or ii) visit same node twice.
Since (0) takes $O(V^2)$, (1) takes $O(1)$ and (2) takes $O(V^2)$, the time complexity of patched version is still $O(VE)$.